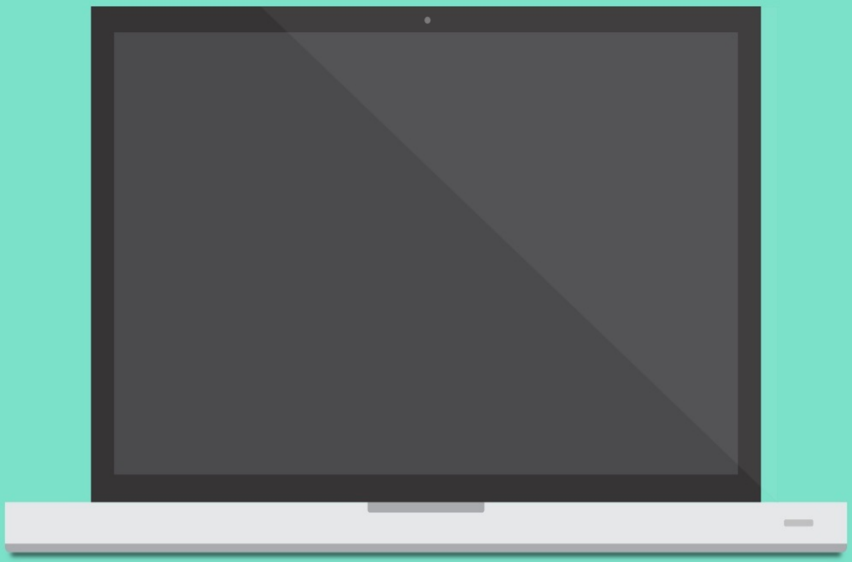

Mac 开发配置手册

by Ashu

目錄

Introduction	0
System Preferences	1
XCode	2
Homebrew	3
Usage	3.1
Cask	3.2
iTerm2	4
ZSH	4.1
Git	5
Git ignore	5.1
MySql	6
Node.js	7
Apps	8

Mac 开发配置手册

手册内容为「如何让一部全新的 **MacBook** 快速完成开发环境配置」，主要面向 Web 开发者。其中的指导，在 Mavericks 和 Yosemite 上有效，其他版本系统并未尝试。

- 如果你是一名老手，本手册让你减少配置开发环境的烦恼。
- 如果你是一名新手，那么恭喜你，你将会认识一个全新的世界。

手册内容主要意译自：Sourabh Bajaj 的 Mac OS X Setup Guide，少部分内容由译者添加和修改。

- 阅读手册
- Github地址
- 联系译者

注：译者更推荐各位阅读 Sourabh Bajaj 的英文手册，将系统语言设置为 English，避免相关术语在记忆、理解和查找时产生混淆。

Mac 开发配置手册
by Ashu

系统设置

在任何的操作系统中，首先你需要做一件事就是更新系统，点击窗口左上角的 **>** 关于本机 **>** 软件更新 。此外，如果这是一部新的电脑，你还需要到系统设置进行一些适当调整。如何调整，取决于个人喜好。

触控板

- 系统设置 > 触控板
  - 光标与点击
    - ✓ 轻拍来点按
    - ✓ 辅助点按
    - ✓ 查找
    - ✓ 三指拖移
  - 滚动缩放
    - ✓ 默认全选
  - 更多手势
    - ✓ 默认全选

Dock

- 置于屏幕上的位置：左边
- 设置 Dock 图标更小（大小随个人喜好）
- ✓ 自动显示和隐藏 Dock

Finder

- Finder > 显示
  - 显示标签页栏
  - 显示路径栏
  - 显示状态栏
  - 自定工具栏 > 去除所有按钮，仅剩搜索栏
- Finder > 偏好设置
  - 通用
    - 开启新 Finder 窗口时打开：HOME「用户名」目录
  - 边栏
    - 添加 HOME「用户名」目录 和 创建代码文件目录
    - 将 共享的(shared) 和 标记(tags) 目录去掉

菜单栏

- 去掉蓝牙等无需经常使用的图标
- 将电池显示设置为百分比

Spotlight

- 去掉字体和书签与历史记录等不需要的内容
- 设置合适的快捷键

互联网帐户

- 添加 iCloud 用户，同步日历，联系人和 Find my mac 等等

XCode

从 App store 或苹果开发者网站安装 Xcode 。

紧接着，安装 Xcode command line tools，运行：

```
xcode-select --install
```

运行命令后，按照指引，你将完成 Xcode command line tools 安装。

`译注`：

如果你不是一名 iOS 或 OS X 开发者，可以跳过安装 XCode 的过程，直接安装 Xcode command line tools 。安装完成后，你将可以直接在 terminal 中使用主要的命令，比如：`make, GCC, clang, perl, svn, git, size, strip, strings, libtool, cpp` 等等。

如果你想了解 Xcode command line tools 包含多少可用的命令，可以到 `/Library/Developer/CommandLineTools/` 查看。以下为其中的命令列表：

- ar
- as
- asa
- bison
- BuildStrings
- c++
- c89
- c99
- cc
- clang
- clang++
- cmpdylib
- codesign_allocate
- CpMac
- cpp
- ctags
- ctf_insert
- DeRez
- dsymutil
- dwarfdump
- dyldinfo
- flex

- flex++
- g++
- gatherheaderdoc
- gcc
- gcov
- GetFileInfo
- git
- git-cvsserver
- git-receive-pack
- git-shell
- git-upload-archive
- git-upload-pack
- gm4
- gnumake
- gperf
- hdxml2manxml
- headerdoc2html
- indent
- install_name_tool
- ld
- lex
- libtool
- lipo
- lldb
- lorder
- m4
- make
- MergePef
- mig
- mkdep
- MvMac
- nasm
- ndisasm
- nm
- nmedit
- otool
- pagestuff
- projectInfo
- ranlib
- rebase
- redo_prebinding
- ResMerger

- resolveLinks
- Rez
- RezDet
- RezWack
- rpcgen
- segedit
- SetFile
- size
- SplitForks
- strings
- strip
- svn
- svnadmin
- svndumpfilter
- svnlook
- svnrdump
- svnserve
- svnsync
- svnversion
- unifdef
- unifdefall
- UnRezWack
- unwinddump
- what
- xml2man
- yacc

Homebrew

包管理工具可以让你安装和更新程序变得更方便，目前在 OS X 系统中最受欢迎的包管理工具是 Homebrew.

安装

在安装 Homebrew 之前，需要将 **Xcode Command Line Tools** 安装完成，这样你就可以使用基于 Xcode Command Line Tools 编译的 Homebrew。

在 terminal 中复制以下命令（不包括 `$` ），跟随指引，将完成 Hombrew 安装。

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install
```

紧接着，我们需要做一件事让通过 Homebrew 安装的程序的启动链接 (在 `/usr/local/bin` 中）可以直接运行，无需将完整路径写出。通过以下命令将 `/usr/local/bin` 添加至 `$PATH` 环境变量中:

```
$ echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bash_profile
```

Cmd+T 打开一个新的 terminal 标签页，运行以下命令，确保 brew 运行正常。

```
$ brew doctor
```

`译注：`

安装完成后，Homebrew 会将本地 `/usr/local` 初始化为 git 的工作树，并将目录所有者变更为当前所操作的用户，将来 `brew` 的相关操作不需要 sudo 。

Homebrew 基本使用

安装一个包，可以简单的运行：

```
$ brew install <package_name>
```

更新 Homebrew 在服务器端上的包目录：

```
$ brew update
```

查看你的包是否需要更新：

```
$ brew outdated
```

更新包：

```
$ brew upgrade <package_name>
```

Homebrew 将会把老版本的包缓存下来，以便当你想回滚至旧版本时使用。但这是比较少使用的情况，当你想清理旧版本的包缓存时，可以运行：

```
$ brew cleanup
```

查看你安装过的包列表（包括版本号）：

```
$ brew list --versions
```

Homebrew Cask

你已经感受到了使用 Homebrew 安装命令行程序的便利。那么接下来，我们将通过 Homebrew Cask 优雅、简单、快速的安装和管理 OS X 图形界面程序，比如 Google Chrome 和 Dropbox。

安装

安装 Homebrew-cask 是如此的简单直接，运行以下命令即可完成：

```
$ brew install caskroom/cask/brew-cask
$ brew cask install google-chrome // 安装 Google 浏览器
$ brew update && brew upgrade brew-cask && brew cleanup // 更新
```

搜索

如果你想查看 cask 上是否存在你需要的 app，可以到 caskroom.io 进行搜索。

文件预览插件

有些 插件 可以让 Mac 上的文件预览更有效，比如语法高亮、markdown 渲染、json 预览等等。

```
$ brew cask install qlcolorcode
$ brew cask install qlstephen
$ brew cask install qlmarkdown
$ brew cask install quicklook-json
$ brew cask install qlprettypatch
$ brew cask install quicklook-csv
$ brew cask install betterzipql
$ brew cask install webpquicklook
$ brew cask install suspicious-package
```

OS X 图形界面程序

```
$ brew cask install alfred
$ brew cask install appcleaner
$ brew cask install cheatsheet
$ brew cask install dropbox
$ brew cask install google-chrome
$ brew cask install onepassword
$ brew cask install sublime-text
$ brew cask install totalfinder
...
```

`译注：` 译者本人并不喜欢 brew cask 的安装方式，更倾向于到 App Store 或官方下载 OS X 图形界面程序。主要因为名字不好记忆、偶尔需要手动更新，另外当你使用 Alfred 或 Spotlight ，你将发现将程序安装在 ~/Application 会很方便。

iTerm2

作为一名开发者，我们常常花上很多时间在终端 上，如同武士的剑，一出手便知高低。所以让我们安装 Mac 上最强大的终端 iTerm2 吧！写码除虫，居家必备。

在 **Finder** 中，将 **iTerm** 拖拽进入 Application 文件夹中。然后，你可以在 **Launchpad** 中启动 iTerm。

颜色和字体设置

- 在 Keys -> Hotkey 中设置 `command + option + i` 快速显示和隐藏 iTerm
- 在 Profiles -> Default -> Check silence bell
- 下载 Solarized dark iterm colors，在 Profiles -> Default -> Colors -> Load Presets 将其导入，作为默认颜色。
- 在 Profiles -> Text 改变游标（cursor）文字和颜色，随个人喜好。
- 更多设置，可参考 打造好用的终端

Zsh

我们将安装 `zsh` ，其拓展功能和主题将由 `oh-my-zsh` 提供。其中 `Env.sh` 文件用于维护别名（aliases），输出（exports）和路径改变（path changes）等等，以免影响 `~/.zshrc` 。

Zsh

使用 Homebrew 完成 zsh 和 zsh completions 的安装

```
brew install zsh zsh-completions
```

安装 oh-my-zsh 让 zsh 获得拓展功能和主题

```
curl -L https://github.com/robbyrussell/oh-my-zsh/raw/master/tools/install.sh | sh
```

用文本编辑器或 vi 打开 `.zshrc` 进行以下编辑:

```
ZSH_THEME=pygmalion
alias zshconfig="vi ~/.zshrc"
alias envconfig="vi ~/Projects/config/env.sh"
plugins=(git colored-man colorize github jira vagrant virtualenv pip python brew osx
```

用文本编辑器或 vi 打开 `~/Projects/config/env.sh` 进行以下编辑:

```
#!/bin/zsh

# PATH
export PATH="/usr/local/share/python:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin"
export EDITOR='vi -w'
# export PYTHONPATH=$PYTHONPATH
# export MANPATH="/usr/local/man:$MANPATH"

# Virtual Environment
export WORKON_HOME=$HOME/.virtualenvs
export PROJECT_HOME=$HOME/Projects
source /usr/local/bin/virtualenvwrapper.sh

# Owner
export USER_NAME="YOUR NAME"
eval "$(rbenv init -)"

# FileSearch
function f() { find . -iname "*$1*" ${@:2} }
function r() { grep "$1" ${@:2} -R . }

#mkdir and cd
function mkcd() { mkdir -p "$@" && cd "$_"; }

# Aliases
alias cppcompile='c++ -std=c++11 -stdlib=libc++'
```

`译注：`

如果是新增环境变量或者是修改环境变量的值，都需要 source 一下才能立即生效。

如果是删除一个环境变量，必须输入 exit 以 logout 当前 shell ，然后再重新打开一个新的 shell 并 login 才能生效。

Git and Github

作为一名开发者怎么可能没有 Git 呢? 我们马上就来安装：

```
$ brew install git
```

好的，现在我们来测试一下 `git` 是否安装完好：

```
$ git --version
```

运行 `$ which git` 将会输出 `/usr/local/bin/git` .

接着，我们将定义你的 Git 帐号（与你在 GitHub 使用的用户名和邮箱一致）

```
$ git config --global user.name "Your Name Here"
$ git config --global user.email "your_email@youremail.com"
```

这些配置信息将会添加进 `~/.gitconfig` 文件中.

我们将推荐使用 HTTPS 方法（另一个是 SSH），将你的代码推送到 Github 上的仓库。如果你不想每次都输入用户名和密码的话，可以按照此 描述 说的那样，运行：

```
$ git config --global credential.helper osxkeychain
```

此外，如果你打算使用 SSH方式，可以参考此 链接.

Git Ignore

创建一个新文件 `~/.gitignore` ，并将以下内容添加进去，这样全部 git 仓库将会忽略以下内容所提及的文件。

```
# Folder view configuration files
.DS_Store
Desktop.ini

# Thumbnail cache files
._*
Thumbs.db

# Files that might appear on external disks
.Spotlight-V100
.Trashes

# Compiled Python files
*.pyc

# Compiled C++ files
*.out

# Application specific files
venv
node_modules
.sass-cache
```

MySQL

安装

我们将使用 Homebrew 安装 MySQL，同时也会安装 MySQL 的相关文件。

安装 MySQL:

```
$ brew update # 这是一个好习惯
$ brew install mysql
```

在使用 MySQL 前，我们需要做一些设置：

```
$ unset TMPDIR
$ mkdir /usr/local/var
$ mysql_install_db --verbose --user=`whoami` --basedir="$(brew --prefix mysql)" --datadir
```

使用

启动 MySQL 服务，运行 `mysql.server`

```
$ mysql.server start
```

关闭 MySQL，运行：

```
$ mysql.server stop
```

你可以了解更多 `mysql.server` 的命令，运行：

```
$ mysql.server --help
```

登录 MySQL, 运行:

```
$ mysql -uroot
```

Note: 默认情况下，MySQL 用户 `root` 没有密码，这对本地开发没有关系，但如果你希望修改密码，你可以运行:

```
$ mysqladmin -u root password 'new-password'
```

译注：

当你在设置密码时出现问题，可以参考 lgn21st 的方式。

此外，如果你觉得敲那么多命令是一件很麻烦的事情，那么你也可以参考 Mac OS安装MySQL（使用二进制PGK包安装）

Node.js

使用 Homebrew 安装 Node.js:

```
$ brew update
$ brew install node
```

一般 Node modules 通常被安装在每个项目的本地文件夹 `node_modules` ，但有几个包推荐你安装在全局：

CoffeeScript、 Less、 Grunt 或 Gulp

```
$ npm install -g coffee-script
$ npm install -g less
$ npm install -g grunt-cli
$ npm install -g gulp
```

Npm 使用

安装包:

```
$ npm install <package>     # 安装在本地项目中
$ npm install -g <package>  # 安装在全局
```

安装包，并且将其保存你项目中的 `package.json` 文件:

```
$ npm install <package> --save
```

查看 npm 安装的内容:

```
$ npm list     # 本地
$ npm list -g  # 全局
```

查看过期的包（本地或全局）:

```
$ npm outdated [-g]
```

更新全部或特别指定一个包:

```
$ npm update [<package>]
```

卸载包:

```
$ npm uninstall <package>
```

Apps

这里推荐的 apps 在开发者圈子内普遍评价不错，能便利的处理日常的开发和使用的任务。以下推荐分为四类：

- 开发者工具
- 生产力工具
- 办公工具
- 其他

Developer Tools

- Google Chrome
- Webstorm
- Sketch
- Dash
- Sequel Pro
- Parallels
- Github

Productivity

- 1Password : 跨平台的密码管理工具

- Alfred 2 : 搜索工具，强烈建议更新至 power pack，可以参考 借助 Alfred 2 的 Workflows 功能可以做哪些好玩的事情？

- AppCleaner: 应用程序卸载工具

- Dropbox: 文件同步工具

- Reeder：RSS 阅读工具

- Pocket : 稍后阅读工具

- Spectacle : 让窗口成比例的显示，在写代码调试的时候很方便

- Unarchiver: 支持多种格式（包括 windows下的格式）的压缩/解压缩工具

- OminiFocus ：时间管理工具

- Mou：Markdown 编辑器，国人出品

- [Sip] 取色器

- [MindeNode] 思维导图

- [xScope] 测量工具

- [Lantern] 科学上网

- [OhMyStar] Github star 管理

Office Apps

- Keynote
- Numbers
- Pages
- Microsoft Office

Others

- CheatSheet : 长按 `command` ，将能查看当前程序的快捷键
- Tweetbot: 最好的 twitter 客户端，优雅，精致